Constrained On-line Adaptation for Aircraft Control

Silvia Ferrari

Advisor: Prof. Robert F. Stengel

Princeton University

FAA/NASA Joint University Program on Air Transportation,

Princeton University, Princeton NJ

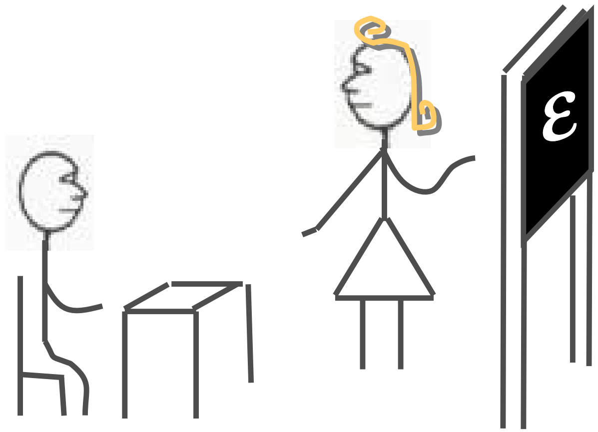
January 10-11, 2002

A Multiphase Learning Approach to Automated Reasoning

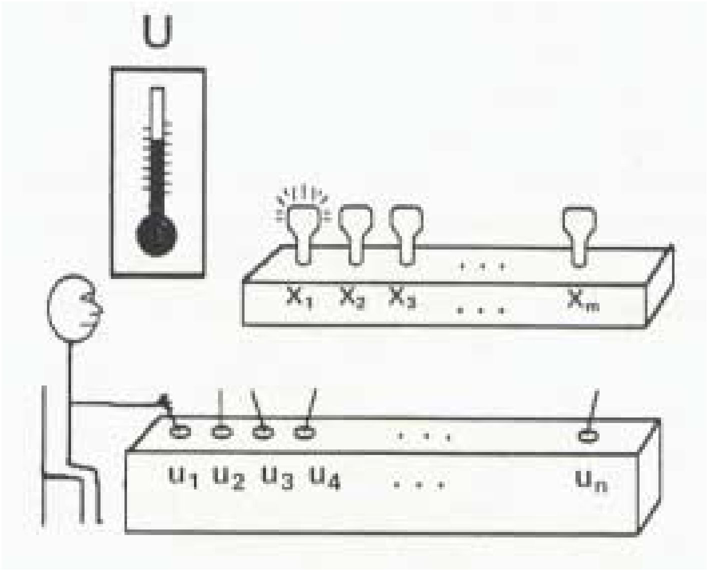
On-line

Control	Routing
Identification	Scheduling
Planning	...

Supervised Learning:



Reinforcement Learning:



Introduction

- Stringent operational requirements introduce
  Complexity
  Nonlinearity
  Uncertainty

- Classical/neural synthesis of control systems
  A-priori knowledge
  Adaptive neural networks

- On-line adaptation takes place during every time interval:

Action network takes immediate control action

Critic network estimates projected cost

Full Envelope Aircraft Control

- Multiphase learning
  Initialization: match linear controllers exactly off-line
  On-line learning: full-scale simulations, testing, or operation

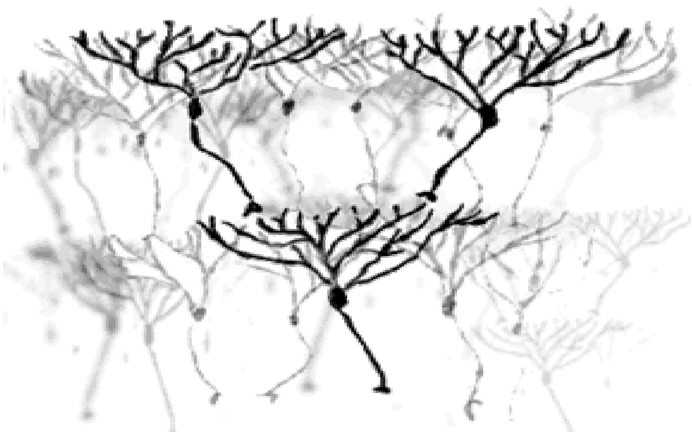
- On-line training improves performance w.r.t. linear controllers:
  Differences between actual and assumed models
  Nonlinear effects not captured in linearizations

- Algebraically constrained on-line adaptation:
  Preserve linear control knowledge

- Potential applications:
  Incorporate pilot's knowledge into controller *a-priori*
  Uninhabited air vehicles control
  Aerobatic flight control

Motivation for Neural Network-Based Controller

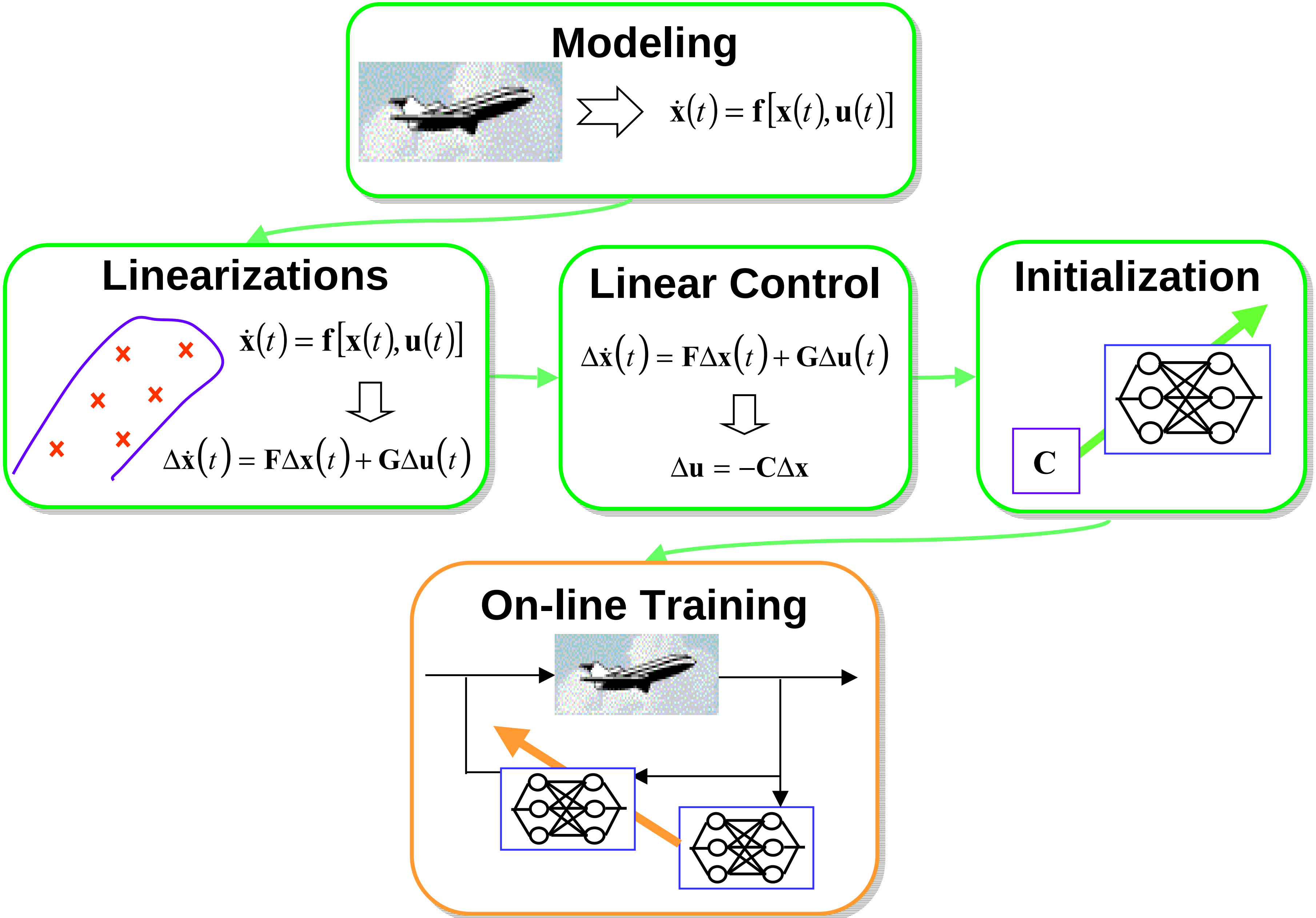
Neural Networks for control: **coping with complexity**



- Learning
- Flexible logic
- Applicability to nonlinear systems
- Applicability to multi-variable systems
- Parallel distributed processing and hardware implementation

Table of Contents

Aircraft Control Design Approach



Linear Control Design

Linearizations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}[\mathbf{x}(t), \mathbf{u}(t), \mathbf{p}(t)]$$

$$\Downarrow$$

$$\Delta\dot{\mathbf{x}}(t) = \mathbf{F}\Delta\mathbf{x}(t) + \mathbf{G}\Delta\mathbf{u}(t)$$

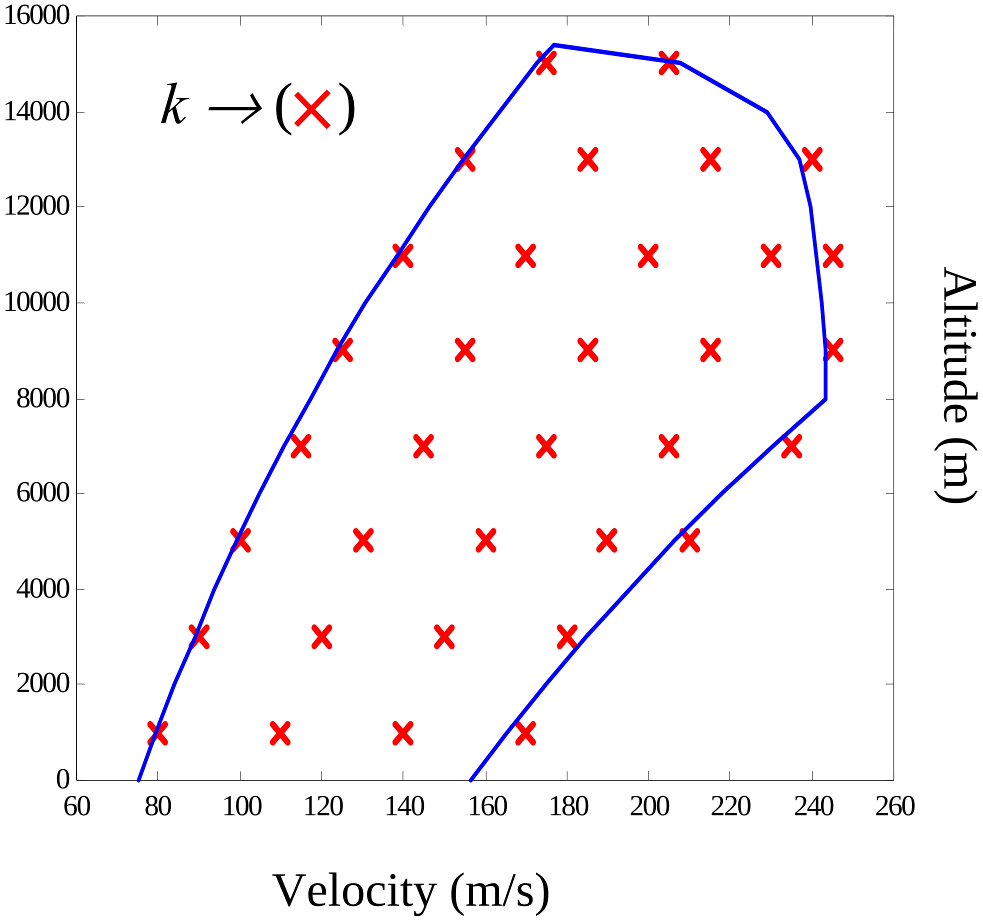
$$\Downarrow$$

$$\begin{cases} \Delta\dot{\mathbf{x}}_L(t) = \mathbf{F}_L\Delta\mathbf{x}_L(t) + \mathbf{G}_L\Delta\mathbf{u}_L(t) \\ \Delta\dot{\mathbf{x}}_{LD}(t) = \mathbf{F}_{LD}\Delta\mathbf{x}_{LD}(t) + \mathbf{G}_{LD}\Delta\mathbf{u}_{LD}(t) \end{cases}$$

Linear control design:

- Longitudinal (L)
- Lateral-directional (LD)

Aircraft Flight Envelope $\{V, H\}$: $(\gamma = \mu = \beta = 0)$



Proportional Integral Linear Control Law

Quadratic cost function to be minimized:

$$J = \lim_{t_f \to \infty} \frac{1}{2} \int_0^{t_f} L[\mathbf{x}_a(\tau), \tilde{\mathbf{u}}(\tau)] d\tau$$

$$= \lim_{t_f \to \infty} \frac{1}{2} \int_0^{t_f} \left[\mathbf{x}_a^T(\tau) \mathbf{Q} \mathbf{x}_a(\tau) + 2\mathbf{x}_a^T(\tau) \mathbf{M} \tilde{\mathbf{u}}(\tau) + \tilde{\mathbf{u}}^T(\tau) \mathbf{R} \tilde{\mathbf{u}}(\tau) \right] d\tau$$

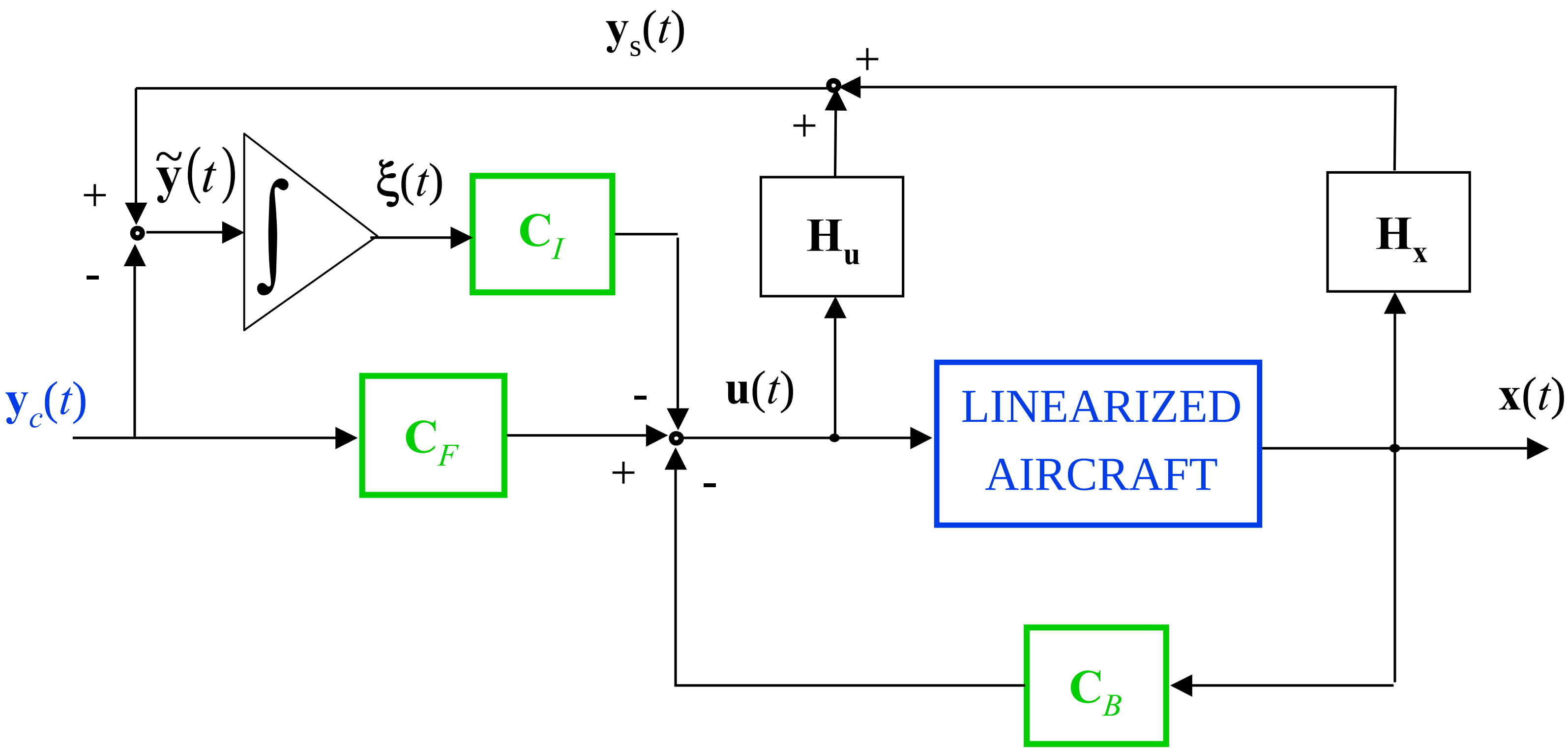
where $\tilde{\mathbf{y}}(t) \equiv \mathbf{y}_S(t) - \mathbf{y}_C(t)$, $\boldsymbol{\xi}(t) = \int_0^t \tilde{\mathbf{y}}(\tau) d\tau$, and $\mathbf{x}_a \equiv \begin{bmatrix} \tilde{\mathbf{x}}^T & \boldsymbol{\xi}^T \end{bmatrix}^T$

***Minimizing Linear Control Law*:**

$$\tilde{\mathbf{u}}(t) = -\mathbf{C}\mathbf{x}_a(t) = -\mathbf{C}_B \tilde{\mathbf{x}}(t) - \mathbf{C}_I \boldsymbol{\xi}(t) \equiv \Delta\mathbf{u}_B(t) + \Delta\mathbf{u}_I(t)$$

Linear Proportional-Integral Controller

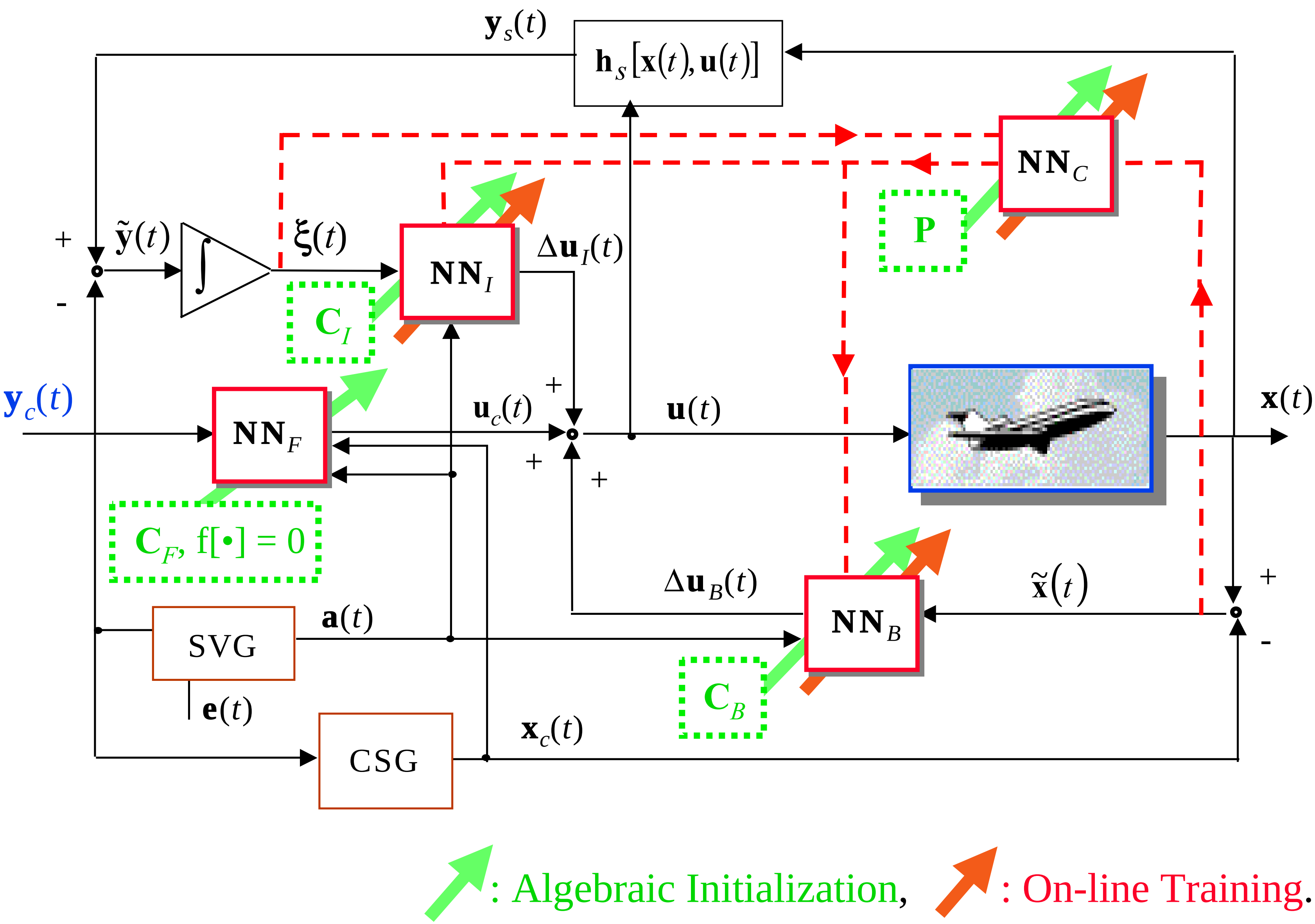
Closed-loop stability: $\mathbf{x} \to \mathbf{x}_c, \quad \mathbf{u} \to \mathbf{u}_c, \quad \tilde{\mathbf{y}} \to 0$



Omitting Δ's, for simplicity:

$\tilde{\mathbf{x}}(t) \equiv \mathbf{x}(t) - \mathbf{x}_c(t), \;\; \tilde{\mathbf{u}}(t) \equiv \mathbf{u}(t) - \mathbf{u}_c(t), \ldots, \;\; \mathbf{y}_c$ = desired output, $(\mathbf{x}_c, \mathbf{u}_c)$ = set point.

Proportional-Integral Neural Network Controller



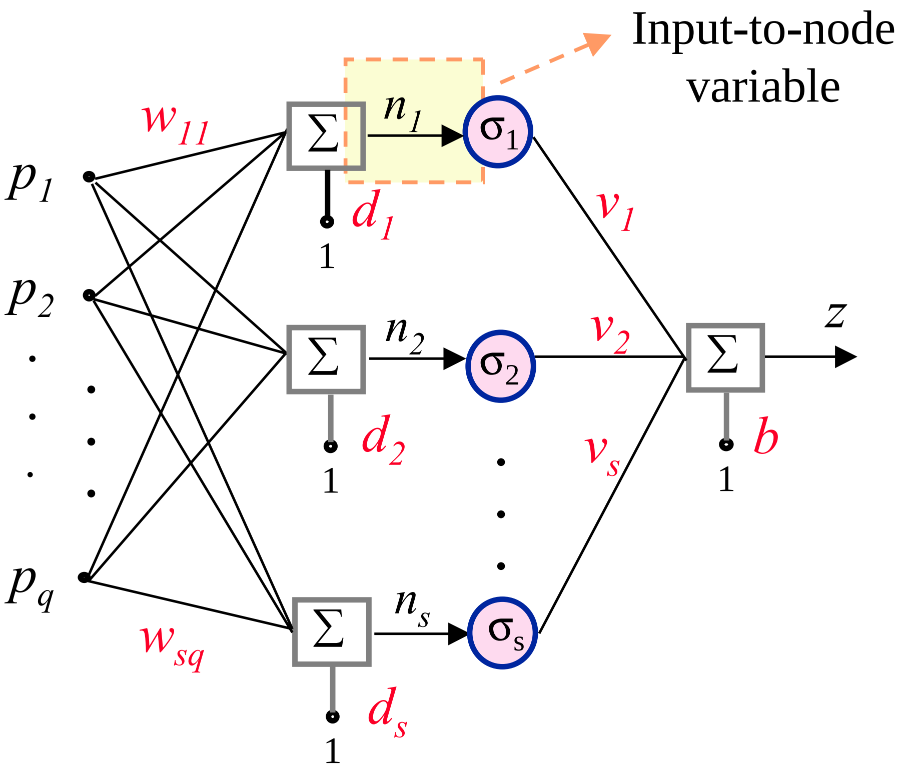
One-hidden Layer Sigmoidal Neural Network

Output: $z = NN(\mathbf{p})$

Input: $\mathbf{p}$

Adjustable parameters:

$\mathbf{W}$, $\mathbf{d}$, $\mathbf{v}$



s - Hidden nodes

Output equations:

$$z = \mathbf{v}^T \boldsymbol{\sigma}[\mathbf{W}\mathbf{p} + \mathbf{d}]$$

Gradient equations:

$$\frac{\partial z}{\partial p_j} = \sum_{i=1}^{s} \frac{\partial z}{\partial n_i} \frac{\partial n_i}{\partial p_j}$$

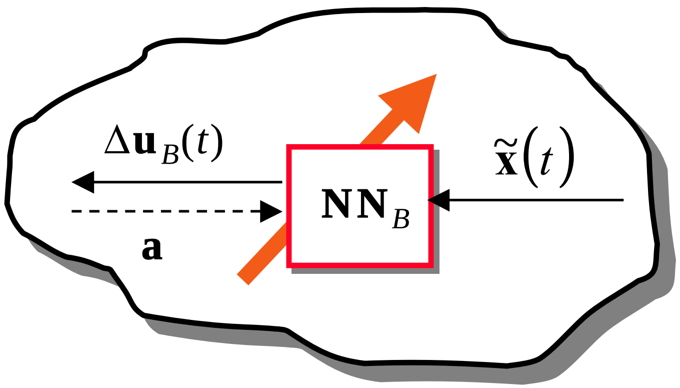
$$= \sum_{i=1}^{s} v_i \sigma'(\underline{n_i}) w_{ij},\ j = 1, \ldots, q$$

$$\sigma(n) = \frac{(e^n - 1)}{(e^n + 1)}, \quad \begin{cases} -\infty < n < \infty \\ -1 < \sigma(n) < 1 \end{cases}$$

Feedback (Action) Neural Network Initialization

From the Proportional-Integral optimal control law:

- $\Delta \mathbf{u}_B\left[\tilde{\mathbf{x}}(t)\right] = -\mathbf{C}_B \tilde{\mathbf{x}}(t) \rightarrow \Delta \mathbf{u}_B[\mathbf{0}] = \mathbf{0}$
- $\dfrac{\partial \Delta \mathbf{u}_B(t)}{\partial \tilde{\mathbf{x}}(t)} = -\mathbf{C}_B$



Feedback Neural Network Initialization Requirements:

Accounts for regulation, $\mathbf{z}_B = \mathbf{NN}_B(\tilde{\mathbf{x}}, \mathbf{a})$. For each operating point, k,

(R1) $\mathbf{z}_B\left(\mathbf{0}_{n\times 1}, \mathbf{a}^k\right) = \mathbf{0}$

(R2) $\left.\dfrac{\partial \mathbf{z}_B(t)}{\partial \tilde{\mathbf{x}}(t)}\right|_{\tilde{\mathbf{x}}=\mathbf{0},\quad \mathbf{a}=\mathbf{a}^k} = \left.\dfrac{\partial\left(\Delta \mathbf{u}_B(t)\right)}{\partial \tilde{\mathbf{x}}(t)}\right|_{\tilde{\mathbf{x}}=\mathbf{0},\quad \mathbf{a}=\mathbf{a}^k} = -\mathbf{C}_B^k$

General Form of Initialization Requirements

Data set: $\{\mathbf{y}^k,\ 0,\ \mathbf{c}^k\}_{k=1,\dots,p}$ Known neural network..

- $\mathbf{c}^k$ → **Gradient**
- 0 → **Output**
- $\mathbf{y}^k$ → **Input**

Specifications:

$$\begin{cases} z\left(\mathbf{y}^k\right) = 0 \\ \dfrac{\partial z}{\partial \mathbf{p}}\left(\mathbf{y}^k\right) = \mathbf{c}^k \end{cases}, \text{ where } \mathbf{y}^k = \begin{bmatrix} \mathbf{x}^k \\ \mathbf{a}^k \end{bmatrix} = \begin{bmatrix} \mathbf{0} \\ \mathbf{a}^k \end{bmatrix}$$

Output and Gradient Nonlinear Transcendental Initialization Equations:

$$0 = \mathbf{v}^T \boldsymbol{\sigma}[\mathbf{W}\mathbf{y}^k + \mathbf{d}]$$

$$(\mathbf{c}^k)^T = \mathbf{W}^T \{\mathbf{v} \otimes \boldsymbol{\sigma}'[\mathbf{W}\mathbf{y}^k + \mathbf{d}]\},\ \ k = 1, \dots, p$$

Algebraic Initialization Principles

If all input-to-node values are assumed known:

$$\mathbf{n}^k \equiv [n_1^k \quad \cdots \quad n_s^k]^T = \mathbf{W}\mathbf{y}^k + \mathbf{d}, \quad k = 1, \ldots, p$$

$$\boxed{\begin{aligned} \mathbf{u} &= \mathbf{S}\mathbf{v} \\ \mathbf{c}^k &= \mathbf{B}^k \mathbf{W} \end{aligned}}$$

Output and Gradient Linear Algebraic Initialization Equations

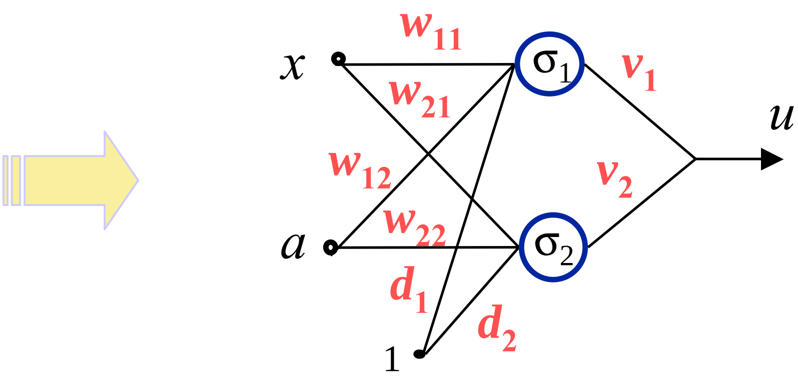
Where:

$$\mathbf{u} \equiv \left[u^1 \quad \cdots \quad u^p\right]^T, \ \mathbf{B}^k \equiv \left\{\mathbf{v} \otimes \boldsymbol{\sigma}'\left[\mathbf{n}^k\right]\right\}^T, \ \mathbf{S} \equiv \begin{bmatrix} \sigma\left(n_1^1\right) & \sigma\left(n_2^1\right) & \ldots & \sigma\left(n_s^1\right) \\ \sigma\left(n_1^2\right) & \sigma\left(n_2^2\right) & \ldots & \sigma\left(n_s^2\right) \\ \vdots & \vdots & \ddots & \vdots \\ \sigma\left(n_1^p\right) & \sigma\left(n_2^p\right) & \ldots & \sigma\left(n_s^p\right) \end{bmatrix}.$$

Algebraic Initialization Example: 2-nodes NN_B

Data Set:

Op. Point	A	B
x^k	0	0
a^k	8	10
u^k	0	0
$c^k = (\partial u/\partial x)^k$	15	-8

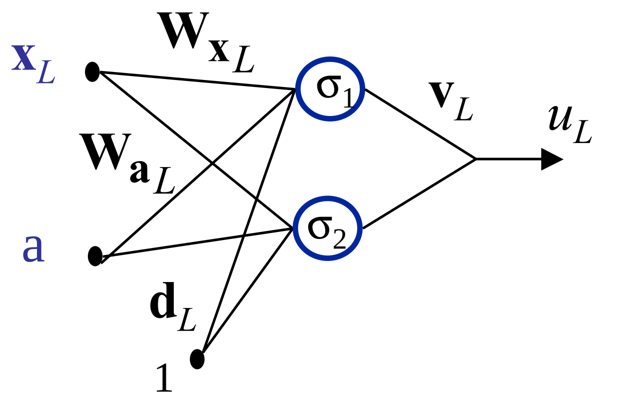


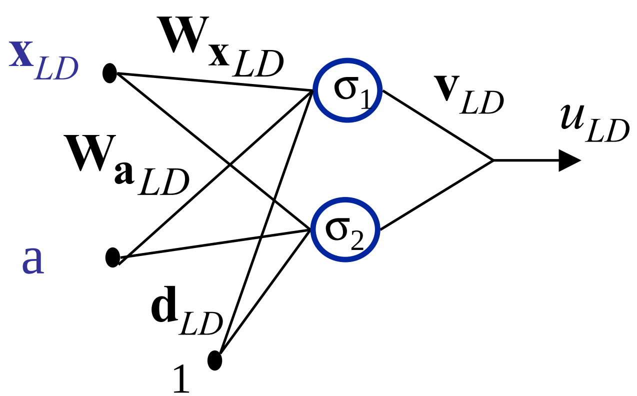
Algebraic Solution:

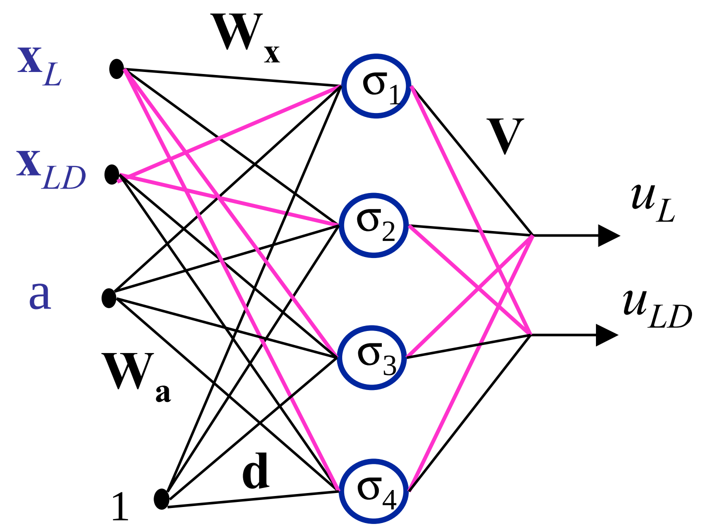
- **Pick any input-to-node values, n_1^A, n_2^A, n_1^B, and n_2^B**
- $u^A = v_1\sigma(n_1^A) + v_2\sigma(n_2^A)$, $u^B = v_1\sigma(n_1^B) + v_2\sigma(n_2^B) \rightarrow v_1, v_2$
- $c^A = v_1\, w_{11}\sigma'(n_1^A) + v_2\, w_{21}\sigma'(n_2^A)$, $c^B = v_1\, w_{11}\sigma'(n_1^B) + v_2\, w_{21}\sigma'(n_2^B) \rightarrow w_{11}, w_{21}$
- $n_1^A = w_{11}\, x_1^A + w_{12}\, x_2^A + d_1$, $n_1^B = w_{11}\, x_1^B + w_{12}\, x_2^B + d_1$, $\rightarrow w_{12}, d_1$
- $n_2^A = w_{21}\, x_1^A + w_{22}\, x_2^A + d_2$, $n_2^B = w_{21}\, x_1^B + w_{22}\, x_2^B + d_2$, $\rightarrow w_{22}, d_2$

Joining Two Initialized Longitudinal and Lateral Neural Networks

Initialized Networks:



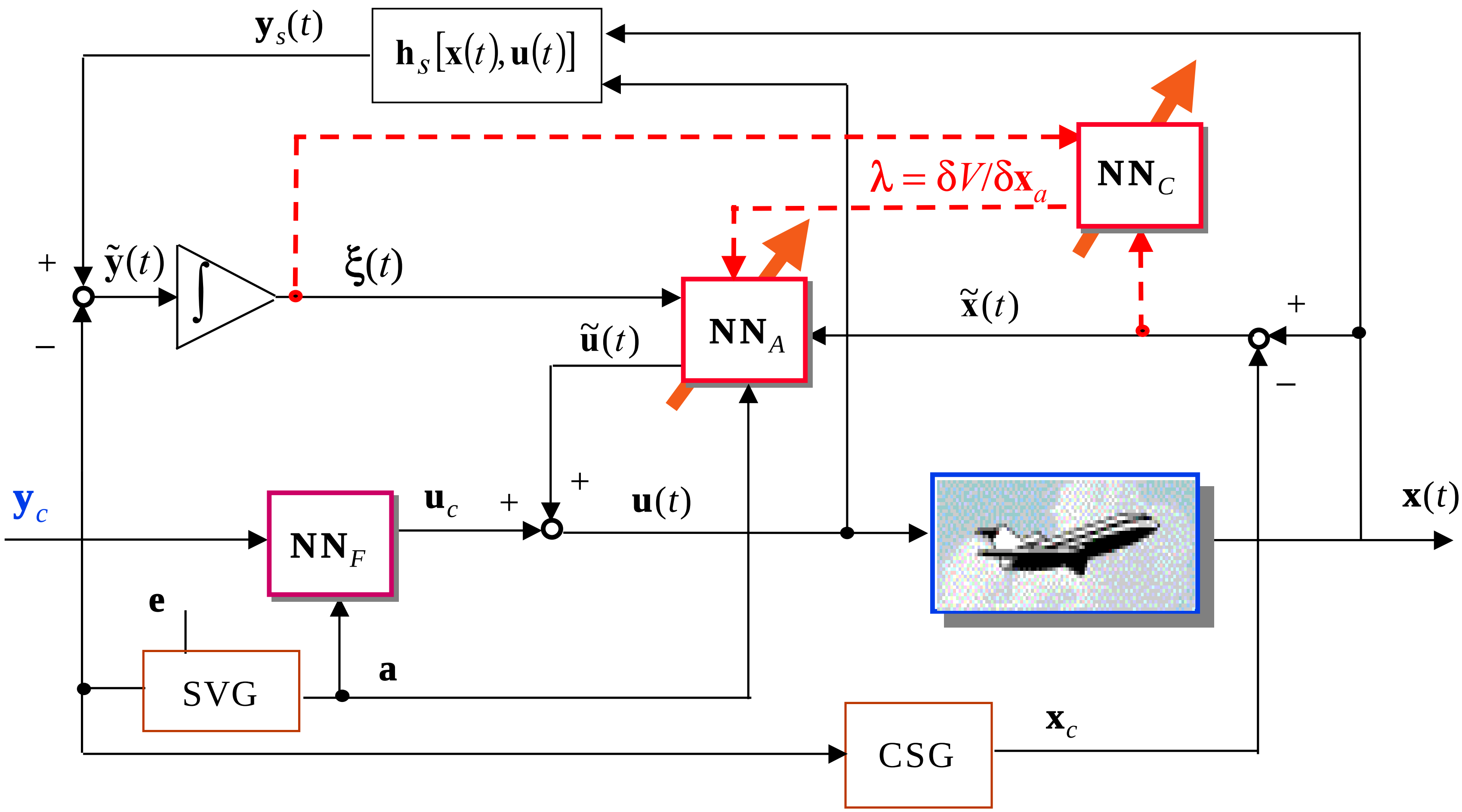




$$\mathbf{W}_\mathbf{x} = \begin{bmatrix} \mathbf{W}_{\mathbf{x}_L} & \mathbf{0} \\ \mathbf{0} & \mathbf{W}_{\mathbf{x}_{LD}} \end{bmatrix} \quad \mathbf{d} = \begin{bmatrix} \mathbf{d}_L \\ \mathbf{d}_{LD} \end{bmatrix}$$

$$\mathbf{W}_\mathbf{a} = \begin{bmatrix} \mathbf{W}_{\mathbf{a}_L} \\ \mathbf{W}_{\mathbf{a}_{LD}} \end{bmatrix} \quad \mathbf{V} = \begin{bmatrix} \mathbf{v}_L^T & \mathbf{0} \\ \mathbf{0} & \mathbf{v}_{LD}^T \end{bmatrix}$$

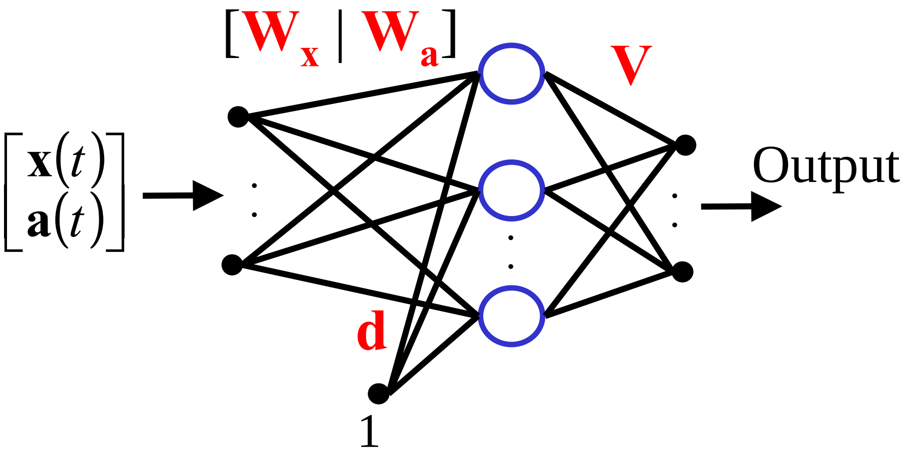
Proportional-Integral Neural Network Controller: On-line Action and Critic Networks Implementation



$$V(t) = -\lim_{t_f \to \infty} \frac{1}{2} \int_{t_f}^{t} \left[\mathbf{x}_a^T(\tau)\mathbf{Q}\mathbf{x}_a(\tau) + 2\mathbf{x}_a^{\tau}(\tau)\mathbf{M}\tilde{\mathbf{u}}(\tau) + \tilde{\mathbf{u}}^T(\tau)\mathbf{R}\tilde{\mathbf{u}}(\tau) \right] d\tau,$$

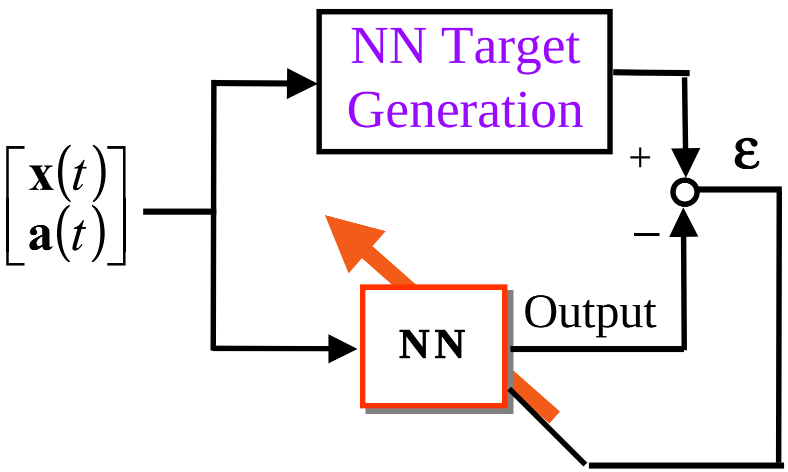
: On-line Training

Action/Critic Network On-line Learning, at Time t



$$\begin{cases} \mathbf{W_x} = \begin{bmatrix} \mathbf{W}_{11} & \mathbf{W}_{12} \\ \mathbf{W}_{21} & \mathbf{W}_{22} \end{bmatrix} \\ \mathbf{w} \equiv [\mathrm{Vec}(\mathbf{W}_{12})^T\ \mathrm{Vec}(\mathbf{W}_{21})^T] \end{cases}$$

Each network must meet its target, subject to <u>Initialization Requirements</u> (IR)

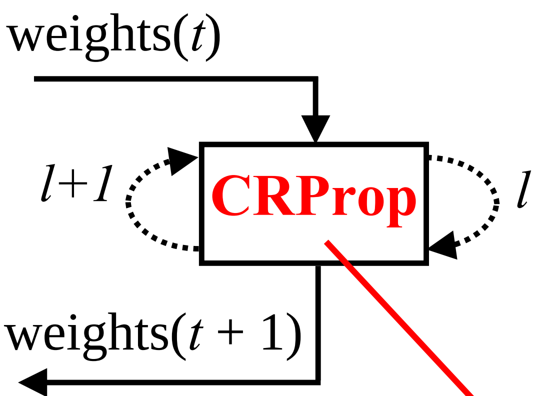


$$\boxed{\min_{\mathbf{W},\mathbf{d},\mathbf{V}} E \equiv \min_{\mathbf{W},\mathbf{d},\mathbf{V}} |\boldsymbol{\varepsilon}|^2, \text{ sbj. to IR}}$$

$$\begin{cases} E \equiv \text{Network performance} \\ \boldsymbol{\varepsilon} \equiv \text{Network error} \\ \mathrm{Vec} \equiv \text{Vec operation} \end{cases}$$

Algebraically Constrained On-line Learning Algorithm

- At time t, the Constrained Resilient Backpropagation (**CRProp**) algorithm minimizes E, computing the weights to be used at $(t + 1)$:



At each epoch, l:

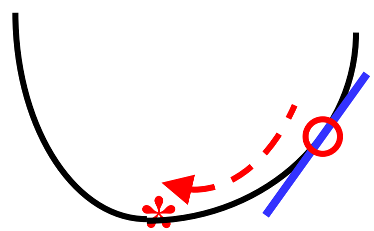
$$\begin{cases} \mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} + \Delta\mathbf{w}^{(l)} \;\rightarrow\; \mathbf{W}_{21}^{(l+1)},\; \mathbf{W}_{12}^{(l+1)} \\ \mathbf{W}_{\mathbf{a}}^{(l+1)} = \mathbf{W}_{\mathbf{a}}^{(l)},\; \mathbf{d}^{(l+1)} = \mathbf{d}^{(l)} \\ \mathbf{v}^{(l+1)} = \mathbf{v}^{(l)} \\ \mathrm{Vec}\left(\mathbf{W}_{11}^{(l+1)}\right) = \mathbf{K}_{11} - \mathbf{K}_{12}\,\mathrm{Vec}\left(\mathbf{W}_{21}^{(l+1)}\right) \\ \mathrm{Vec}\left(\mathbf{W}_{22}^{(l+1)}\right) = \mathbf{K}_{22} - \mathbf{K}_{21}\,\mathrm{Vec}\left(\mathbf{W}_{12}^{(l+1)}\right) \end{cases}$$

Where $\mathbf{K}_{11}$, $\mathbf{K}_{12}$, $\mathbf{K}_{21}$, and $\mathbf{K}_{22}$ are known, constant matrices

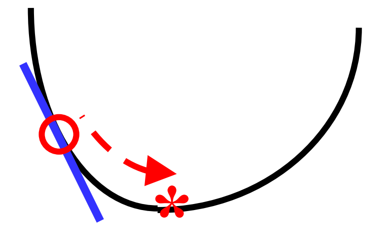
Resilient Backpropagation Algorithm

The size and direction of each weight's increment, $\Delta \mathbf{w}^{(l)}$, are based on the <u>sign</u> of the gradient of the performance, E, w.r.t. the weight, $\mathbf{w}$

Increment Direction:

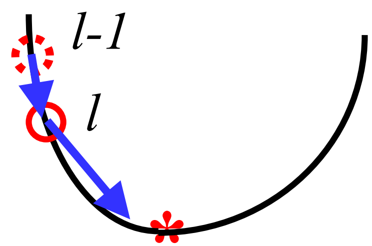


$$\left.\frac{\partial E}{\partial w}\right|^{(l)} > 0$$

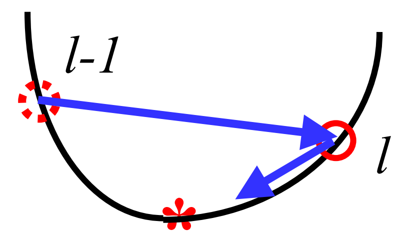


$$\left.\frac{\partial E}{\partial w}\right|^{(l)} < 0$$

Increment Size:



$$\left.\frac{\partial E}{\partial w}\right|^{(l-1)} \left.\frac{\partial E}{\partial w}\right|^{(l)} > 0$$



$$\left.\frac{\partial E}{\partial w}\right|^{(l-1)} \left.\frac{\partial E}{\partial w}\right|^{(l)} < 0$$

Summary and Conclusions

- Objectives:

  Improve performance under unforeseen conditions

  Preserve initialization control knowledge during on-line learning

- Achievements:

  Systematic approach for designing adaptive systems

  Guaranteed fulfillment of adaptation constraints

  Innovative algebraic framework for neural network learning

- Successful implementation of an adaptive critic approach for flight control:
  - Algebraic initialization
  - On-line training by a Resilient Backpropagation algorithm

Other On-line Network-Control Applications:

Process control, air-traffic management, reconfiguring hardware (raw chips), anomaly detection, criminal profiling, image processing, ...